

Everyone Can Code Early Learners



Teacher Guide

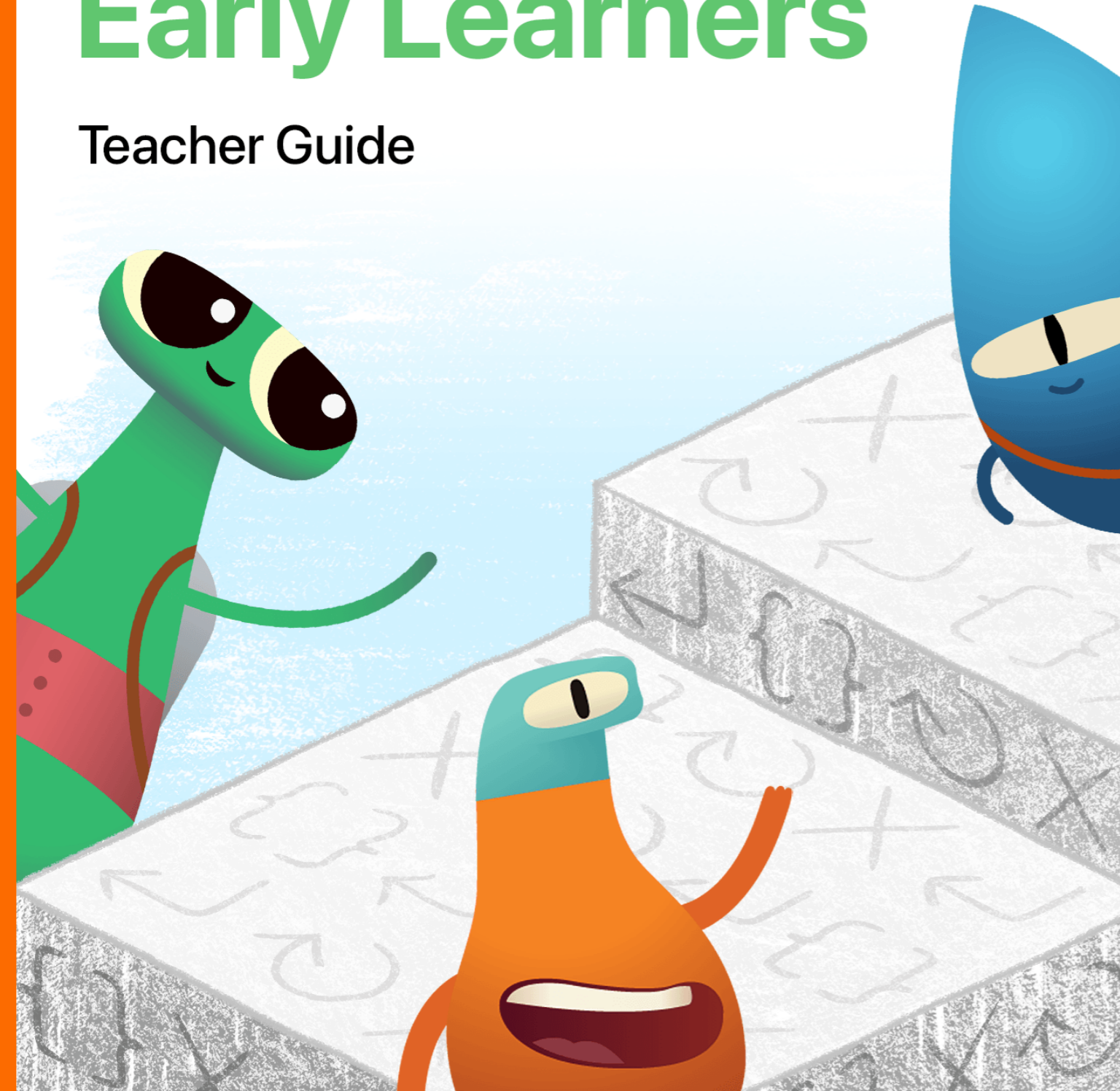


Table of Contents

Introduction

Commands

- Lesson 1: Daily Routines
- Lesson 2: Story Order
- Lesson 3: Dance Moves

Functions

- Lesson 1: Paper Gem
- Lesson 2: Songfest
- Lesson 3: My Calming Function

Loops

- Lesson 1: Repeating Petals
- Lesson 2: Obstacle Course
- Lesson 3: Drumming Patterns

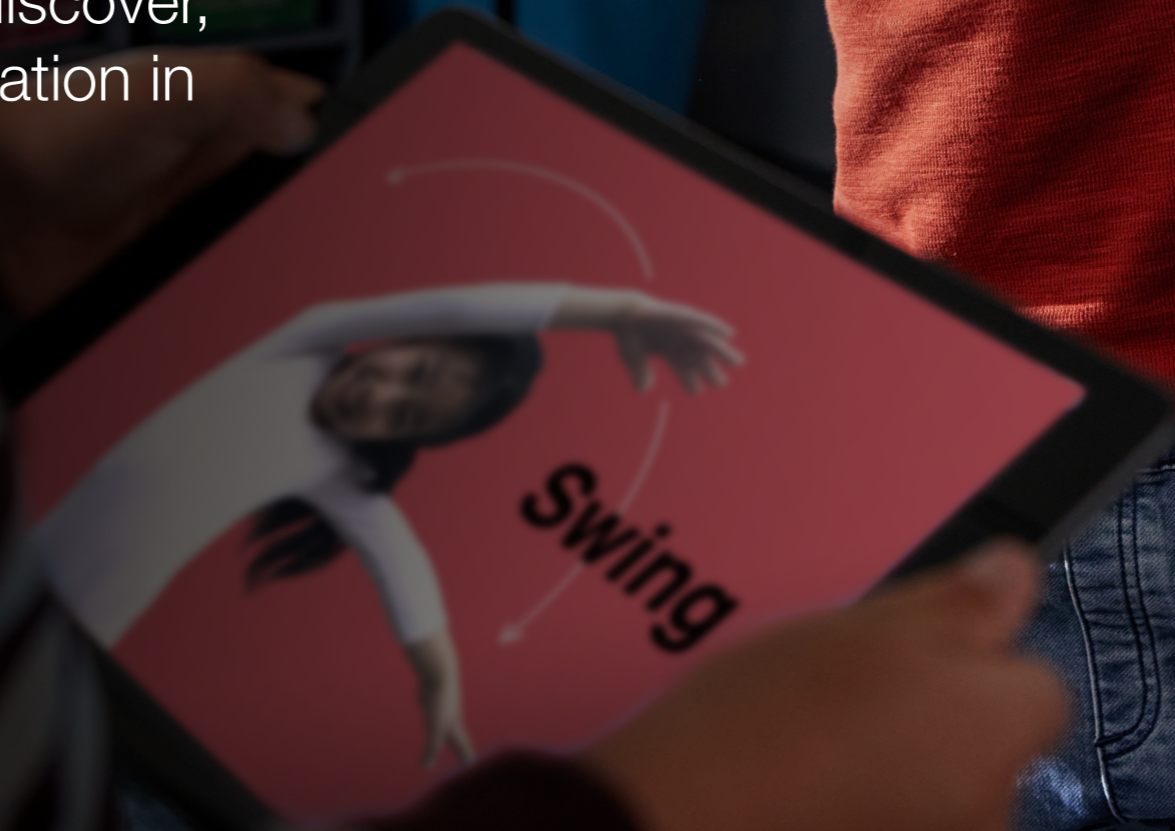
Variables

- Lesson 1: Sink or Float
- Lesson 2: Word Game
- Lesson 3: All About Me

App Design

Facilitator Resources


Everyone Can Code Early Learners is designed to help educators and families introduce coding in the early grades, when learners are first developing computational thinking skills. Through these lessons, learners in kindergarten through third grade will explore, discover, and play to build a foundation in core coding concepts.



Instructional Design

This guide is divided into four modules, as well as a culminating app design project. Every module contains three lessons, each of which focuses on one concept related to coding. Within each lesson, you'll find three activities: Explore, Discover, and Play. The activities can be broken up into multiple sessions or days.

Day 1: Discussion and hands-on learning



Explore

Introduce and discuss the coding concept

~25
minutes

Discover


Build familiarity with the concept through creative activities

Play

- Code along with Byte in the Swift Playgrounds app
- Practice coding in companion worksheets and Keynote activities
- Bring Byte's world into the real world with unplugged floor puzzle coding games

~25
minutes

Day 2: Connecting learning to code



Scope and Sequence

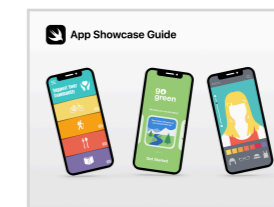
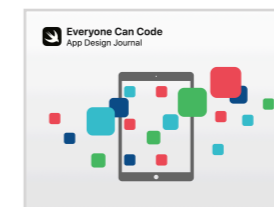
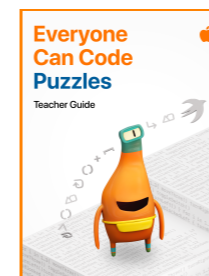
The four modules in this guide are designed to be used across kindergarten to third grade, and they can be done in any order. We encourage you to use the App Design module at any time or even multiple times a year as learners grow their understanding of code and apps.

Example:

| Grade | Module | Culminating project | Approximate total time |
|--------------|---------------------------|----------------------------|------------------------|
| Kindergarten | Commands | App Design | 4 hours |
| First | Functions | App Design | 4 hours |
| Second | Loops | App Design | 4 hours |
| Third | Variables | App Design | 4 hours |

Continue Learning

For teaching fourth to eighth graders, the Everyone Can Code Puzzles, along with the App Design Journal and the App Showcase Guide, offer more than 45 hours of learning. Learn more in the [Everyone Can Code Curriculum Guide](#).



Learner Portfolios (Optional)

Throughout these modules, collect artifacts from the activities to create portfolios with your learners.











| Module | Lesson | Suggested artifacts |
|-------------------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Commands | Daily Routines | <ul style="list-style-type: none"> • Issuing Commands worksheet • Adding a New Command worksheet |
| | Story Order | <ul style="list-style-type: none"> • Story Order plot point picture • Story Order group picture |
| | Dance Moves | <ul style="list-style-type: none"> • Dance Moves cards • Dance Moves video (optional) |
| Functions | Paper Gem | <ul style="list-style-type: none"> • Paper Gem shape • Composing a New Behavior worksheet • Creating a New Function worksheet |
| | Songfest | <ul style="list-style-type: none"> • Songfest concert video or written function |
| | My Calming Function | <ul style="list-style-type: none"> • My Calming Function drawing or video • Collect, Toggle, Repeat worksheet |
| Loops | Repeating Petals | <ul style="list-style-type: none"> • Repeating Petals • Using Loops worksheet • Looping All the Sides worksheet |
| | Obstacle Course | <ul style="list-style-type: none"> • Video or pictures of the obstacle course (optional) |
| | Drumming Patterns | <ul style="list-style-type: none"> • To the Edge and Back worksheet • Video or pictures of the drumming (optional) |
| Variables | Sink or Float | <ul style="list-style-type: none"> • Sink or Float • Keeping Track worksheet |
| | Word Game | <ul style="list-style-type: none"> • Word games |
| | All About Me | <ul style="list-style-type: none"> • All About Me • All About You |
| App Design | | <ul style="list-style-type: none"> • What's an App? • My App Design • App Design prototype |

Getting Started with Swift Playgrounds on iPad or Mac



Before diving into the lessons, be sure that you've downloaded [Swift Playgrounds](#), [Pages](#), and [Keynote](#).

The modules in this guide use different combinations of playgrounds. Here's what you'll need for each module:

| Module | Playgrounds | How to Download in Swift Playgrounds |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Commands |  Learn to Code 1  MeeBot Dances | <p>To subscribe to the MeeBot Playgrounds feed, scroll to the bottom of the More Playgrounds screen and tap Enter a Subscription URL. Then enter: ubtechrobotics.github.io/MeebotPlaygroundFeed/locales.json.</p> |
| Functions |  Learn to Code 1 | |
| Loops |  Learn to Code 1  MeeBot Dances | <p>To subscribe to the MeeBot Playgrounds feed, scroll to the bottom of the More Playgrounds screen and tap Enter a Subscription URL. Then enter: ubtechrobotics.github.io/MeebotPlaygroundFeed/locales.json.</p> |
| Variables |  Learn to Code 2  Rock, Paper, Scissors  Code Machine | <p>Rock, Paper, Scissors and Code Machine can be found in the Books section of the More Playgrounds screen.</p> |
| App Design | | |

Check the minimum requirements for Swift Playgrounds in the [App Store](#). Visit [Apple Support](#) to get help with Swift Playgrounds.

Facilitator Tips

To get the most out of the lessons with your learners, try some of these tips.

Explore and Discover Activities:

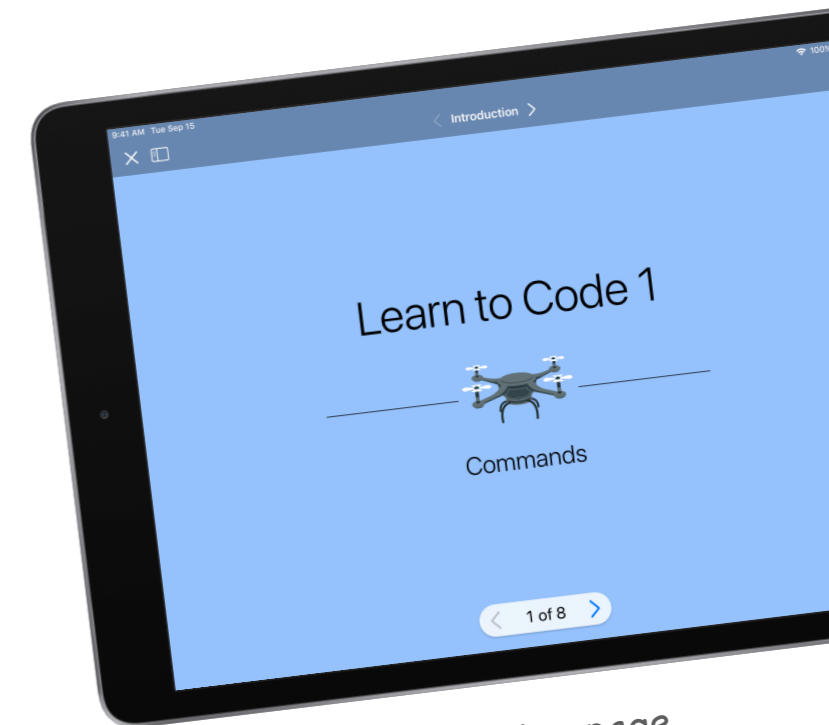
- Simplify any syntax or special casing when writing or showing code — for example:
 - `var names = ["Rose", "Sam", "Joy"]` --> `var names = Rose, Sam, Joy`
 - `var ages = [7, 8, 7, 8, 7]` --> `var ages = 7, 8, 7, 8, 7`
 - `var myFavoriteColor = ■` --> `var my favorite color = ■`

Play Activities:

- To make the Swift Playgrounds app even more simple for your early learners, follow the instructions in the lesson plans. These include:
 - Read the introductions as a whole group
 - Give learners pared-down directions for the accompanying worksheets so they can come up with their own solutions
 - Use one facilitator iPad or Mac to solve the puzzles in the app
- `let` and `var`: The `let` keyword isn't covered in this guide. To avoid confusion in Swift Playgrounds, please change any `let` keywords to `var` before showing the pages to learners. In the playgrounds that we recommend, the two keywords are interchangeable.
 - `let` = variable doesn't change
 - `var` = variable does change

Extensions:

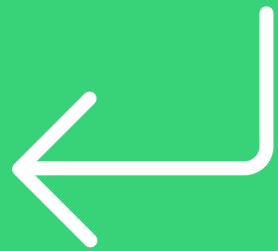
- Expand the floor Play activities to include numeracy, literacy, sight words, spelling, and more. Try the floor Play activity in the Functions module for inspiration.
- Personalize the floor Play activities by having learners make up their own cards for commands, such as `twirl()` or `jump()`.



Introduction page



Playground page



Commands



Overview

Lesson 1: Daily Routines

- Explore: Discussion relating baking to commands
- Discover: Daily Routines activity
- Play: Issuing Commands and Adding a New Command

Lesson 2: Story Order

- Explore: Discussion relating the order of story plots to commands
- Discover: Story Order activity
- Play: Floor puzzle game

Lesson 3: Dance Moves

- Explore: Discussion relating dance moves to commands
- Discover: Dance Moves activity
- Play: Hello MeeBot and Basic Moves

Learners Will Be Able To

- Use everyday examples to describe step-by-step instructions
- Put instructions in order so they make sense
- Test and debug instructions and code

Vocabulary

- **Sequence:** The order in which things happen
- **Step:** One action in a larger process
- **Modify:** To change
- **Command:** Code that tells an application to perform a specific action
- **Bug:** An error in code
- **Debug:** To find and fix errors in code

Standards

1A-AP-08, 1A-AP-10, 1A-AP-12, 1A-AP-14, 1B-AP-16 >

Explore

Objective: Introduce the concept of commands by relating it to baking brownies.

Discussion:

- Would they follow a brownie recipe?
- Would they follow the recipe's steps in order?

Takeaway: Each step or instruction in a recipe is like a command in code. Have learners come up with commands of their own.

Discover

Objective: Model the process of a daily routine by identifying step-by-step instructions.

Materials: Washing Your Hands cards

Directions:

1. Shuffle the deck of Washing Your Hands cards, and lay them out on a table or put them up on the board. The cards should be out of order.
2. Ask learners if they think there's a bug in your handwashing sequence.
3. Ask learners to debug — or fix — the instructions by moving one card at a time to its correct location.

Alternative:

Have learners work in pairs or small groups, and give each group a set of cards.

Extension:

Have learners come up with their own set of step-by-step instructions for something they do every day and make pictures of the specific steps.



[Download the Washing Your Hands cards](#)



Play

Objective: Learners will be able to add the commands in the correct order to collect their first gems in Learn to Code 1 in the Swift Playgrounds app.

Directions:

1. Project the introduction page of the “Commands” chapter in the Learn to Code 1 playground on a screen.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
3. Issuing Commands:
 - Review the two commands that learners will need to get Byte to the gem, `moveForward()` and `collectGem()`.
 - Ask learners to experiment with ways to direct Byte from the start arrow to the gem and collect it. They can record the commands on the worksheet or on a separate piece of paper.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Byte!

Extension:

If learners are ready, move to the next page, Adding a New Command. Here learners will use a new command, `turnLeft()`.



Learn to Code 1

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Learn to Code 1 playground
- Projector or display

Learner materials:

- Issuing Commands and Adding a New Command worksheets
- Pencils
- Extra paper (optional)

↓ [Download the Learn to Code worksheets](#)



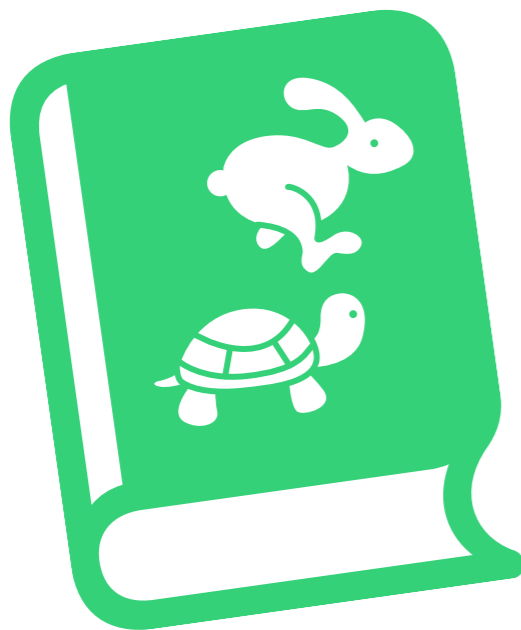
Explore

Objective: Explore how books follow a sequence (beginning to middle to end) in order for the stories to make sense.

Discussion:

- Ask learners if books follow a sequence.
- What would happen if the beginning, middle, and end of a book were out of order?
- Explore several examples.

Takeaway: Make the connection to code, emphasizing how important it is to give coding commands in the correct order, just like the plot points of a story.



Discover

Objective: After creating pictures of various plot points from a story, learners will be able to put the pictures in order to accurately re-create the story.

Facilitator materials:

- Whiteboard
- Markers

Learner materials:

- Paper
- Markers or colored pencils
- Alternative: iPad devices and a drawing app

Directions:

1. Read a story that learners know well. As a class, determine the primary plot points in the story. Ideally, come up with four to six plot points.
2. Create small groups that have the same number of learners as there are plot points — for example, four plot points equals four learners per group.
3. Have each learner in the group draw one of the plot points.
4. Groups take turns standing in front of the room, with learners holding their plot pictures out of order.
5. The audience reorders the pictures, moving one at a time.
6. Take a photo of each group once learners are in the correct order.

Extension or Alternative:

Have each group of learners work on a different story, determining the plot points as a group before drawing the pictures.

Play

Objective: Learners will be able to guide Byte through a physical grid to a gem using directional commands.

Preparation: Learners will be working in groups of three. Use painter's tape to create a four-by-four grid on the floor for each group.

Directions:

1. Distribute the materials, and divide learners into groups of three.
2. Read through each role, and assign each person in the group a role for the first game.
3. Have learners play the game, starting with the designer role.
4. Play three times, each time rotating the role cards.

Roles:

- Designer: Place the gem and the starting arrow on the grid.
- Programmer: With your peers' help, place the command cards on or next to the grid to direct Byte to the gem and collect it.
- Tester: Starting with Byte on the arrow, follow the command cards to move Byte around the grid. If you collect the gem, celebrate! If you don't, work as a team to debug, or fix, the code.

Alternative:

If learners are working with you individually or learning at home, they can play this game on their own using the downloadable alternative Keynote activity.

Facilitator materials:

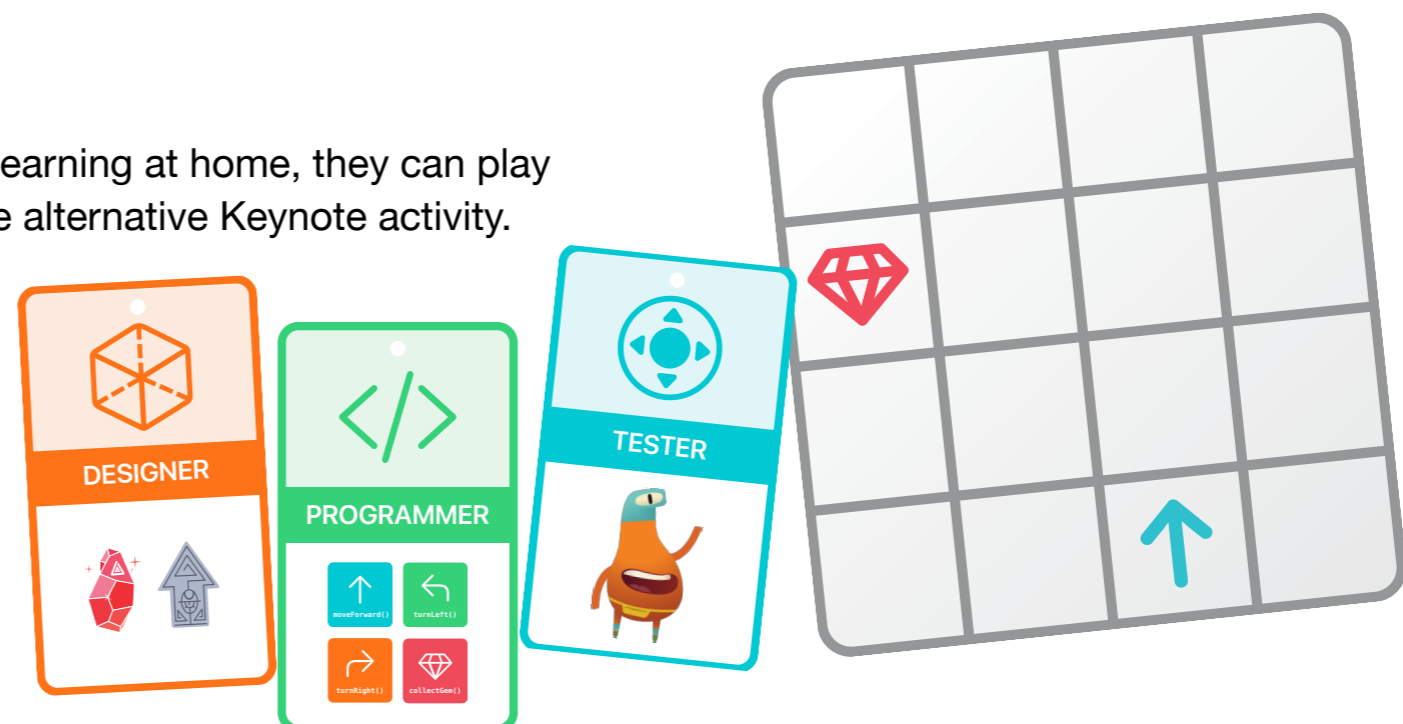
- Painter's tape

Learner materials:

- Role cards
- Command cards: `moveForward()`, `turnLeft()`, `turnRight()`, and `collectGem()`
- Gem
- Byte
- Arrow

↓ [Download the materials](#)

↓ [Download the alternative activity](#)



Explore

Objective: Explore the idea that coding can be creative!

Discussion:

- Ask learners if they've ever learned a dance.
- Did the dance have an order of steps to follow?
- How did they know what to do next?
- Do the dance moves have names?
- Do learners ever use the same moves at different times in a dance, or in different dance routines?

Takeaway: Help learners make the connection that coding is creative and that — like choreographing a dance — coders can make up new commands and then put them together in different and interesting ways.

Discover

Objective: Create a short dance routine, along with cards to represent the dance moves. Each Dance Moves card is like a command in the Learn to Code playground.

Learner materials:

- iPad devices
- Keynote app
- Camera app
- Space to dance

Directions:

1. Have pairs or small groups of learners create a short dance routine.
2. Once learners establish the routine, they'll make cards of the different dance moves. Learners should include a drawing and name of the move on each card, getting as creative and silly as possible.
3. Each group performs their dance — then have a dance party as a whole class!

Alternative:

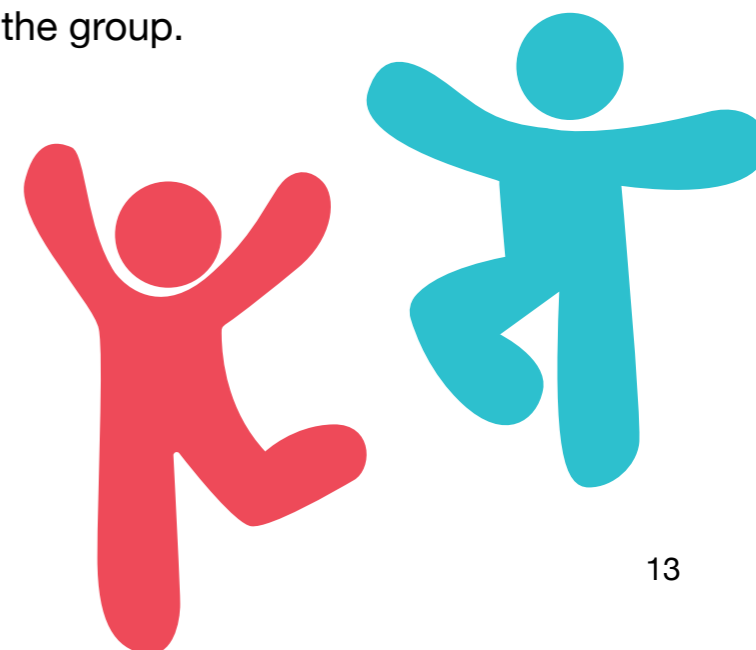
Learners can use the downloadable Dance Moves cards below to create their dance, or they can use the cards as examples when making their own cards.

Extension:

Learners make a video of their dance to show to the group.



[Download the Dance Moves cards](#)



Play

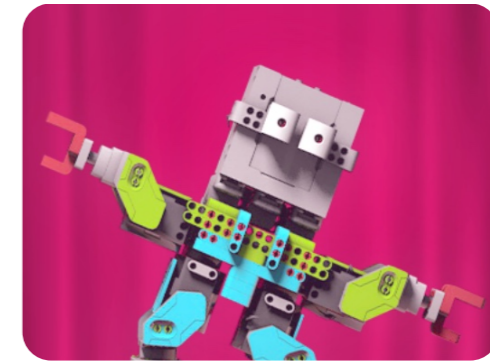
Objective: Create a sequence of steps to teach the MeeBot robot a new dance.

Directions:

1. Project the MeeBot Dances playground on a screen. You'll need to subscribe to the playground if you haven't already.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
3. Hello MeeBot:
 - Click or tap Run My Code, and watch the robot dance.
4. Basic Moves:
 - As a group, in pairs, or individually on their own iPad, learners will choose eight commands from the suggestions list and watch the robot dance.
 - Have learners share their dances, or create a few different dances as a class.
 - Dance with the robot!

Extension:

- Move on to the next page, Dance Routine, where learners can add moves inside the `myDanceRoutine()` function — as many or as few commands as they want.



MeeBot Dances

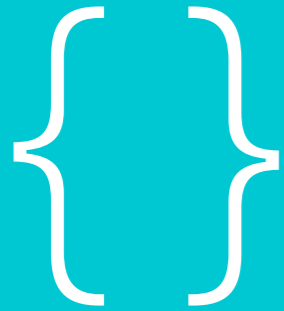
Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- MeeBot Dances playground
- Projector or display

Learner materials:

- iPad devices (optional)





Functions



Overview

Lesson 1: Paper Gem

- Explore: Discussion about step-by-step instructions
- Discover: Paper Gem activity
- Play: Composing a New Behavior and Creating a New Function

Lesson 2: Songfest

- Explore: Discussion about how to name a function
- Discover: Songfest activity
- Play: Floor puzzle game

Lesson 3: My Calming Function

- Explore: Discussion about solving problems in multiple ways
- Discover: My Calming Function activity
- Play: Collect, Toggle, Repeat

Learners Will Be Able To

- Deconstruct a large problem or task into smaller steps
- Create a series of steps to solve a problem or complete a task
- Name functions
- Test and debug code

Vocabulary

- **Function:** A named set of commands that can be run whenever needed
- **Toggle:** To switch on or off

Standards

1A-AP-08, 1A-AP-10, 1A-AP-11, 1A-AP-12, 1A-AP-14, 1B-AP-16 >

Explore

Objective: Explore the idea of packaging a series of commands and giving it a name.

Discussion: Decide on a daily routine to focus on as a class. Have learners identify the name of their daily routine and the steps that make it up.

Example: Bedtime routine

- Step 1: Brush teeth
- Step 2: Use the restroom
- Step 3: Read
- Step 4: Say goodnight
- Step 5: Turn out the lights

Takeaway: Coming up with a set of instructions and giving it a name is the same concept as creating a function.

Extension: Ask learners if the instructions for any of their steps could be more specific. For example, what are the specific steps to brushing your teeth?

Discover

Objective: Learners will begin by following directions to make a paper gem, then they'll write or draw the directions for making another shape of their choice.

Learner materials:

- Paper
- Scissors
- Pencils
- iPad devices (optional)

Directions:

Show learners how to make a paper gem:

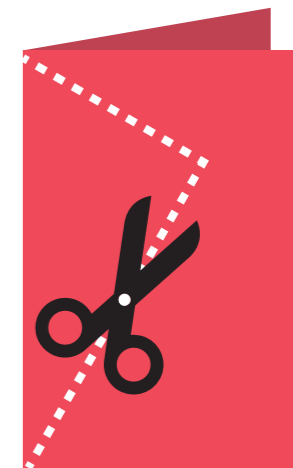
1. Fold a piece of paper in half.
2. Draw a line from the top corner on the folded side to an inch or two above the center of the paper.
3. Draw another line from where the first line ends to the bottom corner on the folded side.
4. Cut along the lines you drew.
5. Remove the gem from the scrap paper, and unfold it.

Ask learners to make their own shapes:

1. Break learners into small groups.
2. Have groups decide on a shape to make.
3. Allow time for learners to practice making the shape once or twice.
4. Have learners write or draw the directions for making the shape, then give their directions a name, such as "Make a Circle" or "The Letter T."

Alternative:

Make a video showing how to make their shapes.



Play

Objective: Working as a whole group, learners will be able to break down the steps needed to get Byte to the gem.

Directions:

1. Project the Learn to Code 1 playground on a screen. Navigate to the “Functions” chapter in Learn to Code 1.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
3. Composing a New Behavior:
 - Review the commands `moveForward()`, `turnLeft()`, and `collectGem()` — keeping in mind that you don’t have a `turnRight()` command.
 - Have learners experiment with ways to direct Byte from the start arrow to the gem and collect it. They record the commands on the worksheet or on a separate piece of paper.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Byte!
4. Creating a New Function:
 - Based on what they learned in the last Playground page, Composing a New Behavior, have learners come up with ideas to create `turnRight()` function.
 - Using their `turnRight()` function, have learners experiment with ways to direct Byte from the start arrow to the closed switch and toggle it.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Byte — that was a hard puzzle!



Learn to Code 1

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Learn to Code 1 playground
- Projector or display

Learner materials:

- Composing a New Behavior and Creating a New Function worksheets
- Pencils
- Extra paper (optional)



[Download the Learn to Code worksheets](#)

Explore

Objective: Apply knowledge of commands and functions to songs by giving them descriptive names.

Discussion: Have learners come up with a variety of songs and give each one a descriptive function name.

Example: For the song “Twinkle, Twinkle, Little Star,” the function call might be `singTwinkle()`, but `singSong1()` wouldn’t be a good name because the first song could change.

Takeaway: Naming functions with descriptive names is important because it makes code easier for you and others to understand.

Discover

Objective: Learners will create a concert by calling different song commands in a concert function.

Facilitator materials:

- iPad or Mac
- Projector or display
- Whiteboard
- Markers

Directions:

1. Help learners create function names for several songs — for example, `singHappyBirthday()`.
2. As a group, choose the order in which to sing the songs.
3. Write a function definition for a concert, and fill in the function with the song commands.

Example:

```
func createConcert() {  
    singHappyBirthday()  
    singTwinkleTwinkle()  
    singMaryHadALittleLamb()  
}  
createConcert()
```

Alternative:

Learners sing in small groups, with each group coming up their own list of songs, song function names, and order to sing the songs in. Each group then performs their songs and makes a video of their concert.

Play

Objective: Learners will solve a simple equation, place a gem on the answer, then guide Byte through the grid using directional commands.

Preparation: Learners will be working in groups of three. Use painter's tape to create a four-by-four grid on the floor for each group. Place the starting arrow inside one square, and place one number inside each remaining square.

Directions:

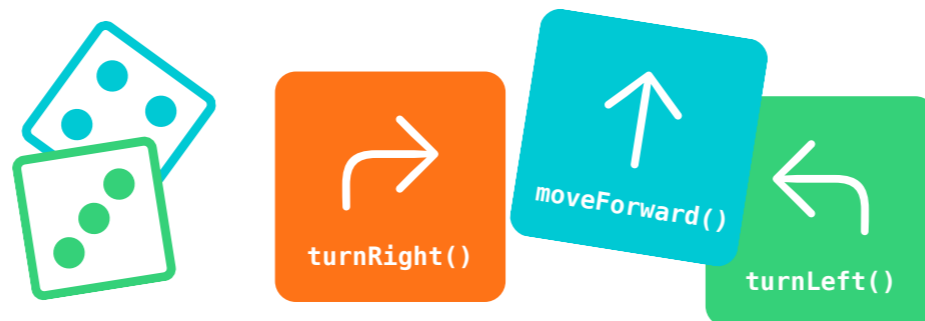
1. Distribute the materials, and divide learners into groups of three.
2. Read through each role, and assign each person in the group a role for the first game.
3. Have learners play the game, starting with the designer role.
4. Play three times, each time rotating the role cards.

Roles:

- Designer: Roll two dice. With the help of your peers, add the two numbers together and place the gem on a grid square that has the sum.
- Programmer: With your peers' help, place the command cards on or next to the grid to direct Byte to the gem and collect it.
- Tester: Starting with Byte on the arrow, follow the command cards to move Byte around the grid. If you collect the gem, celebrate! If you don't, work as a team to fix the code.

Alternative:

If learners are working with you individually or learning at home, they can play this game on their own using the downloadable alternative Keynote activity.



Facilitator materials:

- Painter's tape
- One set of printed numbers for each grid

Learner materials:

- Role cards
- Command cards: `moveForward()`, `turnLeft()`, `turnRight()`, and `collectGem()`
- Gem
- Byte
- Arrow
- Two dice



[Download the materials](#)



[Download the alternative activity](#)

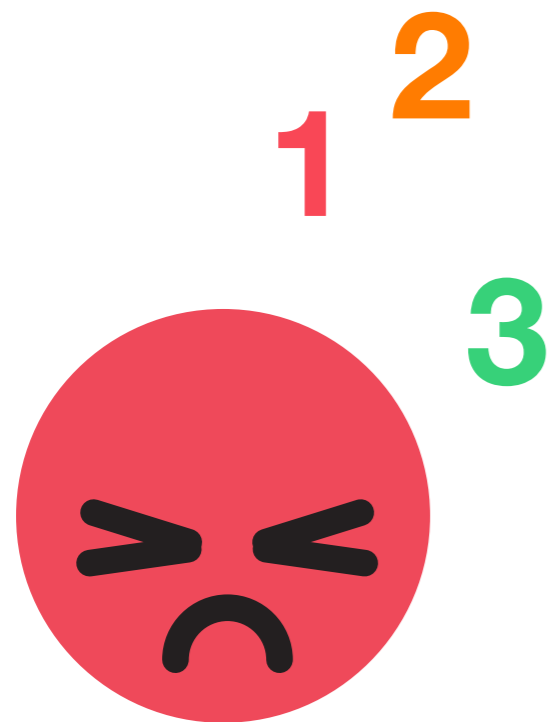


Explore

Objective: Learners will understand that there's usually more than one way to solve a problem.

Discussion: Have learners think about a problem they've had, then share the ways they solved it. Ask the group if anyone would solve that problem in a different way. Explore several different problems and solutions.

Takeaway: Help learners make the connection to code and learn that there's usually more than one way to solve a programming problem.



Discover

Objective: Learners will write a function for their calming technique and give it a name.

Learner materials:

- My Calming Function worksheet
- Pencils
- Colored pens or pencils

Directions:

Tip: It's best that learners work on this activity individually, if possible.

1. Ask learners to brainstorm ways they calm themselves down either at home or at school when they're upset. Have them break down their calming techniques into steps.
2. Distribute the My Calming Function worksheet, and ask learners to draw the steps of their calming technique.
3. Have learners give their calming technique a name. They can use camel case — for example, `countToTen()` — or just use a short sentence, such as “Count to ten.”

Extensions:

Unplugged: Have learners act out their calming technique in small groups or in front of the class.

With iPad: Learners make a video of their calming technique to share with the class.



[Download the My Calming Function worksheet](#)

Play

Objective: Learners will be able to write a function composed of several different types of commands, then use that function to complete a puzzle.

Directions:

1. Project the Collect, Toggle, Repeat page in the Learn to Code 1 playground on a screen, pointing out the empty function that learners will help complete.
2. Collect, Toggle, Repeat:
 - Review the commands `moveForward()`, `turnLeft()`, `turnRight()`, `collectGem()`, and `toggleSwitch()`.
 - Have learners try to identify the parts of the puzzle that repeat, then use their ideas to complete the function in the app and give it a name.
 - Have learners invent a symbol for the function and record the symbol and function name in the commands key on the worksheet.
 - With the additional command, learners experiment with ways to direct Byte to collect all gems and toggle all switches. They record the commands on the worksheet or on a separate piece of paper.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different solutions.
 - Celebrate as a class — this was a hard puzzle!



Learn to Code 1

Facilitator materials:

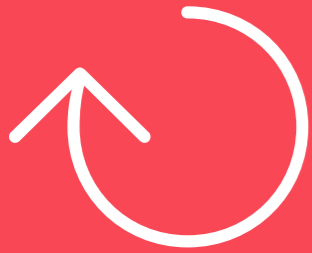
- iPad or Mac
- Swift Playgrounds app
- Learn to Code 1 playground
- Projector or display

Learner materials:

- Collect, Toggle, Repeat worksheet
- Pencils
- Extra paper (optional)

 [Download the Learn to Code worksheet](#)





Loops



Overview

Lesson 1: Repeating Petals

- Explore: Discussion relating repeating steps in code to real life
- Discover: Repeating Petals activity
- Play: Using Loops and Looping All the Sides

Lesson 2: Obstacle Course

- Explore: Discussion about stopping points in a loop
- Discover: Obstacle Course activity
- Play: Floor puzzle game

Lesson 3: Drumming Patterns

- Explore: Discussion about loops in music
- Discover: Drumming Patterns activity
- Play: To the Edge and Back and Dance Loops

Learners Will Be Able To

- Identify a loop in code
- Deconstruct a large problem or task into smaller steps
- Create a sequence of commands and repeat that sequence using a loop
- Test and debug instructions and code

Vocabulary

- **Loop:** A block of code that repeats a certain number of times

Standards

1A-CS-01, 1A-AP-08, 1A-AP-10, 1A-AP-11, 1A-AP-12, 1A-AP-14 >

Explore

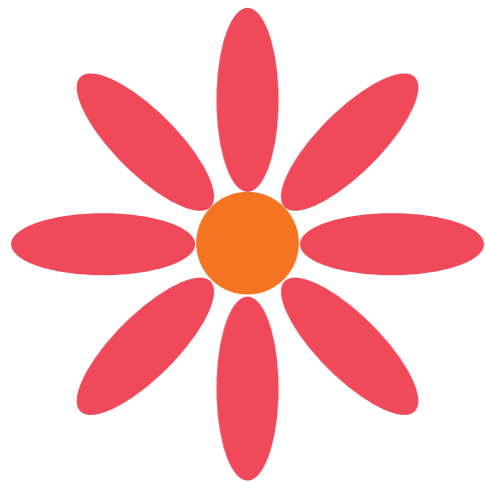
Objective: Connect the idea of loops to real life.

Discussion: Explore times that learners might repeat a task or step in real life.

Examples:

- Walking
- Biking
- Sewing, knitting, or crocheting

Takeaway: Loops repeat a command or a set of commands as many times as you specify.



Discover

Objective: Learners will begin to explore the concept of loops by making a unique flower.

Learner materials:

- Repeating Petals worksheet
- Colored paper
- Pencils
- Scissors
- Glue sticks
- Dice

Directions:

1. Learners draw a single petal — about the length of the palm of their hand — on a piece of colored paper and cut it out. This will be their petal template for their flower.
2. Each learner then rolls two dice, adds the numbers together, and fills in the missing number in the loop on their Repeating Petals worksheet. This is the number of petals that their flower will have.
3. Using their petal template, learners trace their petal on colored paper and cut out the correct number of petals for their flower.
4. Using the Repeating Petals worksheet, learners assemble their flower and glue the parts in place.



[Download the Repeating Petals worksheet](#)



Play

Objective: Learners will be able to write code inside a loop to collect all the gems.

Directions:

1. Project the “For Loops” chapter introduction page in the Learn to Code 1 playground on a screen.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
3. Using Loops:
 - Show learners how portals work, and review the commands `moveForward()`, `turnLeft()`, `turnRight()`, and `collectGem()`.
 - Have learners experiment with ways to direct Byte from the start arrow to the gems and collect them, noticing which commands repeat. They record the commands on the worksheet or on a separate piece of paper.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to direct Byte to collect the first gem and walk to the portal.
 - Ask learners how many gems there are, and add that number to the loop. Click or tap Run My Code.
 - Try several different solutions.
 - Celebrate with Byte!
4. Looping All the Sides:
 - Have learners experiment with ways to collect all the gems, noticing which commands repeat.
 - To add a `for` loop, either use the code suggestions at the bottom of the editor or tap `+` at the top of the screen.
 - Gather ideas from the class, and write the code in Swift Playgrounds to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Byte!




Learn to Code 1

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Learn to Code 1 playground
- Projector or display

Learner materials:

- Using Loops and Looping All the Sides worksheets
- Pencils
- Extra paper (optional)

 [Download the Learn to Code worksheets](#)

Explore

Objective: Explore why loops always need a specific end point.

Discussion: Ask learners to imagine a Ferris wheel or another ride they're familiar with. What would happen if the operator didn't press the button to stop the ride after five rounds? Have learners come up with other examples of what would happen if a loop isn't stopped.

Takeaway: Help learners understand that if they don't put a stop on a loop, it will repeat infinitely.

Discover

Objective: Learners will discover how loops work by looping through an obstacle course that they design.

Materials:

- Space to do physical activity
- Obstacle course props
- Die

Directions:

1. Create a short obstacle course either in your classroom or outside.
2. Roll a die, and have learners repeat the course as many times as the die says.

Alternative:

Learners come up with a series of moves — for example, touch your toes, jump, kick out one leg. Roll a die, and learners repeat the series of moves the number of times on the die.



Play

Objective: Learners will be able to create a puzzle that has a repeating pattern, then solve the puzzle as a group.

Preparation: Learners will be working in groups of three. Use painter's tape to create a four-by-four grid on the floor for each group.

Directions:

1. Distribute the materials, and divide learners into groups of three.
2. Read through each role, and assign each person in the group a role for the first game.
3. Have learners play the game, starting with the designer role.
4. Play three times, each time rotating the role cards.

Roles:

- Designer: With the help of your peers, place three gems in a repeating pattern on the grid. Place the starting arrow on the grid.
- Programmer: With your peers' help, place the command cards on or next to the grid to direct Byte to the gems and collect them. Use the Loop cards to tell the tester how many times to loop through the commands.
- Tester: Starting with Byte on the arrow, follow the command cards to move Byte around the grid. If you collect all the gems, celebrate! If you don't, work as a team to fix the code.

Alternative:

If learners are working with you individually or learning at home, they can play this game on their own using the downloadable alternative Keynote activity.

Facilitator materials:

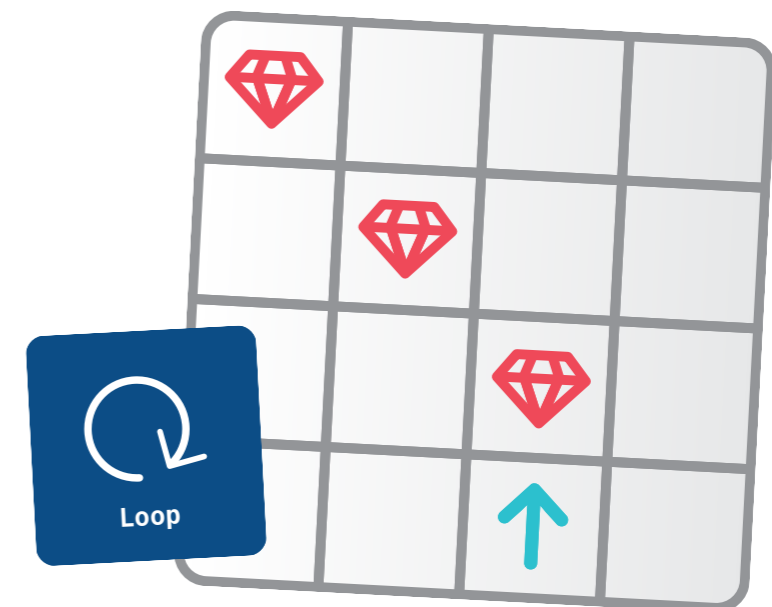
- Painter's tape

Learner materials:

- Role cards
- Command cards: `moveForward()`, `turnLeft()`, `turnRight()`, `collectGem()`, and `Loop`
- Gems
- Byte
- Arrow

↓ [Download the materials](#)

↓ [Download the alternative activity](#)



Explore

Objective: Explore repeating patterns in music.

Discussion: Have learners share about instruments they play or songs they've sung. Ask if they ever repeat a beat or chorus when playing or singing. Can they think of other parts of a song or music that repeat?

Takeaway: Reinforce the idea that loops consist of two parts:

- The commands
- The number of times to repeat

Discover

Objective: Learners will be able to repeat a drum pattern, making a connection between looping code and a real-life physical example.

Materials:

- Something to drum on, such as the floor, thighs, or books
- Space to sit in a circle

Directions:

1. Have learners sit in a circle.
2. Tell learners to repeat the drumbeat that you create, as many times as the number of fingers you hold up. For example, if you hold up four fingers, learners should repeat the drumbeat four times and then stop.
3. Take turns going around the circle or break into small groups so that each learner gets a chance to be the lead drummer.

Extension:

Have learners make drums.



Play

Objective: Learners will call several different commands inside a loop and determine how many times the loop should be called.

Directions:

1. Project the Learn to Code 1 playground on a screen. Navigate to the third page, To the Edge and Back, of the “For Loops” chapter.
2. To the Edge and Back:
 - Review the commands `moveForward()`, `turnLeft()`, `turnRight()`, `collectGem()`, and `toggleSwitch()`.
 - Have learners experiment with ways to direct Byte from the start arrow to each closed switch and toggle it.
 - To add a for loop, either use the code suggestions at the bottom of the editor or tap + at the top of the screen.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Byte!
3. Leave Learn to Code 1 and open the MeeBot Dances playground, navigating to the Dance Loops page. (There’s no learner worksheet for this playground page.)
4. Dance Loops:
 - Have learners work as a group, in pairs, or individually on their own iPad to complete the loop and watch the robot dance.
 - Ask learners to share their dances, or create a few different ones as a class.
 - Dance with the robot!



Learn to Code 1



MeeBot Dances

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Learn to Code 1 playground
- MeeBot Dances playground
- Projector or display

Learner materials:

- To the Edge and Back worksheet
- Pencils
- iPad devices (optional)
- Extra paper (optional)



[Download the Learn to Code worksheet](#)



Variables



Overview

Lesson 1: Sink or Float

- Explore: Discussion about updating a variable
- Discover: Sink or Float activity
- Play: Keeping Track and Sample Game

Lesson 2: Word Game

- Explore: Discussion about types of answers to questions
- Discover: Word Game activity
- Play: Floor puzzle game

Lesson 3: All About Me

- Explore: Discussion about answering questions with lists
- Discover: All About Me activity
- Play: Using a Loop

Learners Will Be Able To

- Associate a variable name with a given value
- Change the value assigned to a variable
- Understand the different Swift types you can assign to a variable, including true/false (Booleans), numbers (Ints), words (Strings), colors (color literals), and images (image literals)
- Test and debug instructions and code

Vocabulary

- **Variable:** A named container that stores a value and can be changed
- **Data:** Information
- **Boolean:** A type that has a value of either true or false

Standards

1A-AP-09, 1B-AP-09, 1B-AP-10, 1B-AP-16 >

Explore

Objective: Explore the concept of variables by counting objects and updating the variable number.

Facilitator materials:

- Whiteboard
- Marker
- Eraser
- Container
- Five pencils (or any five of the same object)

Directions:

1. Begin by writing a variable statement on the whiteboard to keep track of your objects.
 - Example: `var numberOfPencils = 0`
2. Hold up an empty container and tell learners that the container represents your variable, `numberOfPencils`.
3. Add one pencil to the container, and ask learners what the variable count is now. When they answer correctly, erase the `0` and write `1`.
4. Continue until you've added all the pencils and your code reads: `var numberOfPencils = 5`.
5. Then begin taking pencils out of the container, updating the variable as you remove them.

Takeaway: Help learners understand that variables store a bit of information. In this case, the information is a number and the number tells you how many pencils are in the container.

Discover

Objective: Using found objects, learners will conduct experiments to determine if items sink or float, then they'll record the data using images (image literals) and true/false values (Booleans).

Learner materials:

- iPad devices
- Keynote app
- Sink or Float worksheet
- Bucket of water
- Several objects to test

Directions:

1. Break learners into small groups.
2. Have them collect various items to test.
3. For each item, ask learners to:
 - Take a picture of the item and add it to the worksheet.
 - Test the item in the water.
 - Record results on the worksheet by circling either `true` or `false`.



[Download the Sink or Float worksheet](#)

Play

Objective: Learners will be able to create and update variables in two different coding contexts.

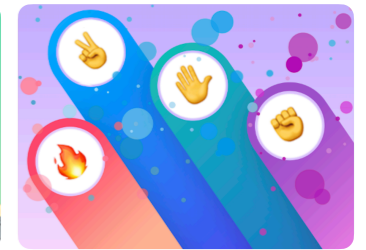
Directions:

1. Project the Learn to Code 2 playground on a screen. Navigate to the “Variables” chapter.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
3. Keeping Track:
 - Have learners experiment with ways to direct Hopper from the start arrow to the gem and collect it. They record the commands on the worksheet or on a separate piece of paper.
 - Gather ideas from the class, and write the code in the Swift Playgrounds app to complete the puzzle. Click or tap Run My Code.
 - Try several different ideas.
 - Celebrate with Hopper!
4. Leave Learn to Code 2 and move on to the last page of the Rock, Paper, Scissors playground, called Sample Game. (There’s no learner worksheet for this playground page.)
5. Sample Game:
 - Click or tap Run My Code to play the game before changing anything.
 - Decide as a group which parts of the game you want to customize. Some fun things you can change include `game.roundsToWin`, `game.challenger.emoji`, `game.addOpponent`, and `game.roundPrize`.
 - Play the game several times, changing something different each time.

Extension: Many variables are established in the Game.swift file. If learners are curious why some variables don’t have `var` in front of them, open the Game.swift file to show them where the game properties were created.



Learn to Code 2



Rock, Paper,
Scissors

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Learn to Code 2 playground
- Rock, Paper, Scissors playground
- Projector or display

Learner materials:

- Keeping Track worksheet
- Pencils
- Extra paper (optional)



[Download the Learn to Code worksheet](#)

Explore

Objective: Explore various answer types in the real world and relate them to various Swift types, including yes/no or true/false (Booleans), numbers (Ints), words (Strings), colors (color literals), and images (image literals).

Facilitator materials:

- Whiteboard
- Markers

Discussion: Come up with some questions as a class that require different types of answers, and write them on the board.

Examples:

- What color are your eyes? → color
- Do you have a pet? → yes/no
- Do you have siblings? → yes/no
- How old are you? → number
- What's your name? → word

Takeaway: Explain that variables also have different types, including numbers, words, colors, pictures, and yes or no answers. Depending on how you create a variable, you'll have to maintain the same type, even if you update the variable to something new. For example, `var myAge = 8` can change to 9, but it can't change to "nine".

Discover

Objective: Learners will be able to complete a word game by filling in the correct answer type.

Learner materials:

- Word Game worksheets
- Pencils
- Colored pencils

Directions:

Have learners work through one or more word games in small groups. Ideally each group should have at least one reader or supporting person. If all learners are nonreaders, do a few games together as a whole group.

Extension: If learners are able to, have them create a word game for a partner to fill out. Encourage them to use numbers, words, colors, images, and yes or no answers for the blanks.



[Download the Word Game worksheets](#)

Play

Objective: Learners will be able to guide Byte to collect several gems, add each gem to a container, and update a variable.

Preparation: Learners will be working in groups of three. Use painter's tape to create a four-by-four grid on the floor for each group.

Directions:

1. Distribute the materials, and divide learners into groups of three.
2. Read through each role, and assign each person in the group a role for the first game.
3. Have learners play the game, starting with the designer role.
4. Play three times, each time rotating the role cards.

Roles:

- Designer: Place multiple gems and the starting arrow on the grid.
- Programmer: With the help of your peers, place the command cards on or next to the grid to direct Byte to the gems and collect them.
- Tester: Starting with Byte on the arrow, follow the commands to move Byte around the grid, adding the gems to the container as you collect them. If you collect all the gems, update the variable `numberOfGems` on the container and celebrate! If you don't collect them all, work as a team to fix the code.

Alternative:

If learners are working with you individually or learning at home, they can play this game on their own using the downloadable alternative Keynote activity.

Facilitator materials:

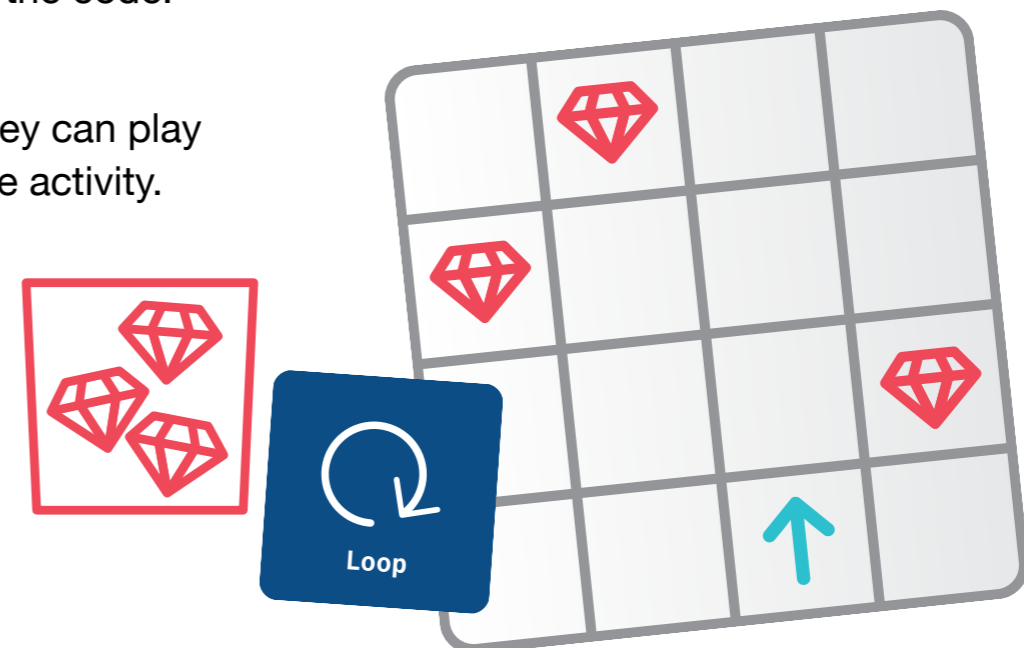
- Painter's tape

Learner materials:

- Role cards
- Command cards: `moveForward()`, `turnLeft()`, `turnRight()`, `collectGem()`, and `Loop`
- Gems
- Byte
- Arrow
- Container labeled:
`var numberOfGems = _____`
- Pen

↓ [Download the materials](#)

↓ [Download the alternative activity](#)









Explore

Objective: Explore how to use lists — or *arrays* — when creating variables.

Discussion: What would happen if a worksheet asked learners for their sibling's name and they have more than one? Collect ideas from the class. If they suggest making a list, tell them that's what coders do! When a variable has more than one answer, learners should create a list.

Have learners come up with questions that could have multiple answers.

Examples:

- Friends' names —> Rose, Sam, Joy
- Learners' ages —> 7, 8, 7, 8, 7, 8, 9, 7, 8, 9, 8
- Favorite colors —>  ,  ,  ,  , 
- Favorite animals —>  ,  ,  , 

Takeaway: Lists that learners create in code are just like lists in a sentence.

Discover

Objective: Learners will be able to fill in variables to describe things about themselves and a partner. Learners might have the opportunity to use an array as a variable type.

Learner materials:

- All About Me and All About You worksheets
- Pencils
- Colored pencils

Directions:

1. Have learners complete the All About Me worksheet.
 - If learners have more than one sibling or pet, have them make a list of items separated by commas.
2. Pair learners up to complete the All About You worksheet.

Alternative: Learners can use their iPad and Keynote to complete the worksheet, taking photos for the picture answers and coloring the color literals using the formatting options.



[Download the All About worksheets](#)

Play

Objective: Learners will be able to identify a variable in code and explore ways they can use arrays with loops.

Directions:

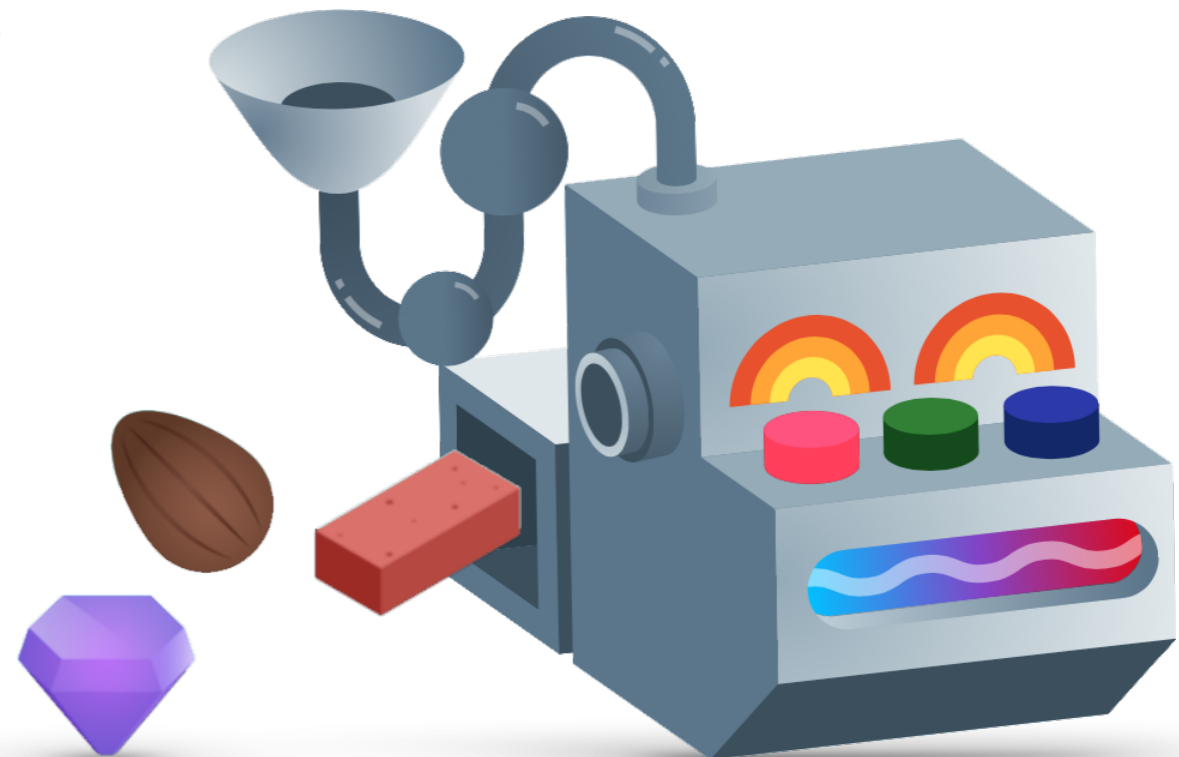
1. Project the Code Machine playground on a screen.
2. Introduction:
 - Read through the pages as a class, stopping for questions if needed.
 - Optional: Play through the first two pages, Exploring the Machine and Combining with Colors.
3. Using a Loop:
 - On this page, learners will combine their knowledge of loops with variables.
 - See if learners can identify the variable in the code that uses an array.
 - Click or tap Run My Code to see what the machine creates.
 - Move on to the second step in the instructions and update the code to include a second variable, items, and a nested loop. Click or tap Run My Code again to see what the machine creates.
 - Note: Try this page before doing the activity with learners.



Code Machine

Facilitator materials:

- iPad or Mac
- Swift Playgrounds app
- Code Machine playground
- Projector or display



App Design



Explore

Objective: Explore familiar apps on various devices.

Directions: Start a discussion about apps that learners use on iPad at home or school. Then talk about apps that they or their parents or guardians use on devices at home.

Takeaway: Reinforce the idea that apps aren't just on phones but also on watches, tablets, computers, and even TV.

Extension: Go deeper into a few examples of apps, asking learners who the app is designed for, what it does, and why they think it was made.

Example:

- App: Swift Playgrounds
- Who it's for: People who want to learn about Swift
- What it does: Helps people learn how to code through puzzles and lessons
- Why it was made: To teach people with little or no programming knowledge how to code

Discover

Objective: Prepare learners to design their own apps by analyzing a familiar app.

Learner materials:

- iPad devices
- What's an App? worksheet
- Pencils
- Colored pens or pencils

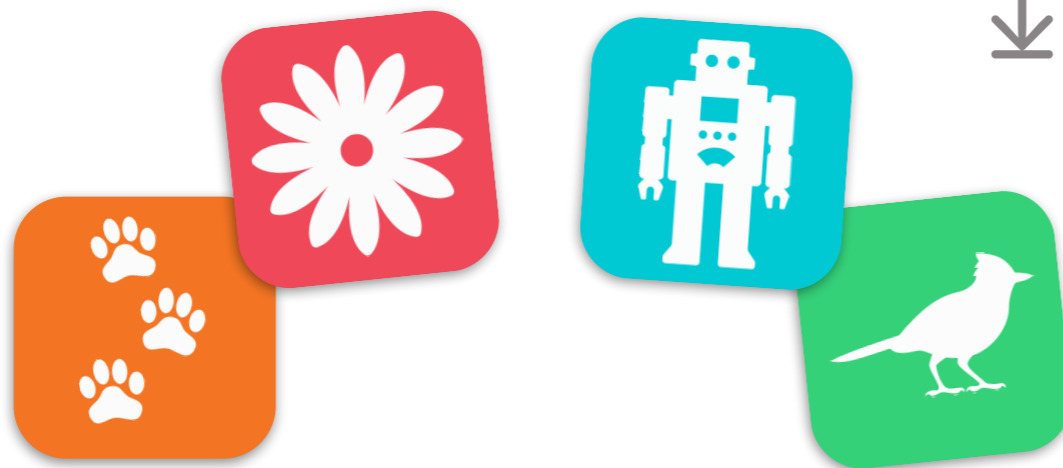
Directions:

1. Break learners into small groups, or have them work individually.
2. Ask learners to choose an iPad app.
3. Have them use the What's an App? worksheet to guide their app exploration.
4. Invite learners to share their findings about the app, either with the whole group or with partners.

Facilitator tip: The younger your learners are, the more help they'll need to complete this worksheet. For a K–1 group, consider doing two or three apps as a whole group.



[Download the What's an App? worksheet](#)



Play

Objective: Learners design their own apps!

Learner materials:

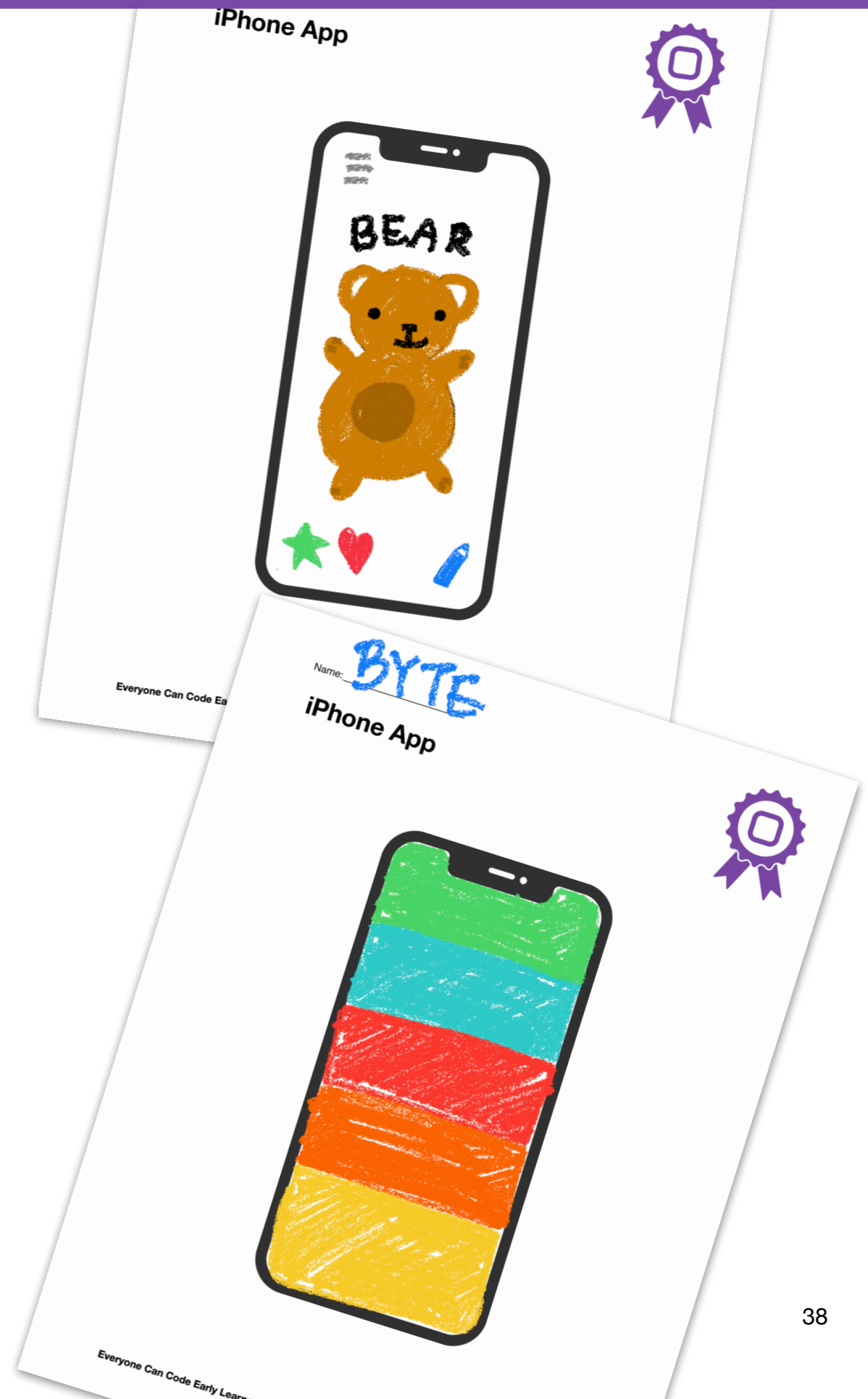
- My App Design worksheet
- Device templates
- Extra paper
- Pencils
- Colored pens or pencils

Directions:

1. Break learners into small groups, or have them work individually.
2. Walk learners through the My App Design worksheet to guide their initial process of designing an app.
3. Have learners prototype the pages of their apps using extra paper or the device templates.
4. Instruct learners to create a final version of their app prototypes using the device templates.
5. Invite each learner or group of learners to present their ideas to the whole group.

↓ [Download the My App Design worksheet](#)

↓ [Download the device templates](#)



Facilitator Resources



Glossary

- **Boolean:** A type that has a value of either true or false
- **Bug:** An error in code
- **Command:** Code that tells an application to perform a specific action
- **Data:** Information
- **Debug:** To find and fix errors in code
- **Function:** A named set of commands that can be run whenever needed
- **Loop:** A block of code that repeats a certain number of times
- **Modify:** To change
- **Sequence:** The order in which things happen
- **Step:** One action in a larger process
- **Toggle:** To switch on or off
- **Variable:** A named container that stores a value and can be changed

CSTA Standards >

1A-AP

- 1A-AP-08: Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
- 1A-AP-09: Model the way programs store and manipulate data by using numbers or other symbols to represent information.
- 1A-AP-10: Develop programs with sequences and simple loops to express ideas or address a problem.
- 1A-AP-11: Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
- 1A-AP-12: Develop plans that describe a program's sequence of events, goals, and expected outcomes.
- 1A-AP-14: Debug (identify and fix) errors in an algorithm or a program that includes sequences and simple loops.

1A-CS

- 1A-CS-01: Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use.

1B-AP

- 1B-AP-09: Create programs that use variables to store and modify data.
- 1B-AP-10: Create programs that include sequences, events, loops, and conditionals.
- 1B-AP-16: Take on varying roles, with facilitator guidance, when collaborating with peers during the design, implementation, and review stages of program development.

Example Answers

These next few pages provide one possible solution for each Swift Playgrounds puzzle — but the puzzles can be solved in more than one way. Encourage learners to try different ways they can direct Byte or other characters.

Celebrate all types of coding and goals that learners might have. Some learners might want to explore all the puzzle space in addition to collecting the gems, while others might want to spin as many times as possible on the way to collecting gems. Don't forget — coding should be fun!



Learn To Code 1

Commands chapter

Issuing Commands

```
moveForward()
moveForward()
moveForward()
collectGem()
```

Commands chapter

Adding a New Command

```
moveForward()
moveForward()
turnLeft()
moveForward()
moveForward()
collectGem()
```

Functions chapter

Composing a New Behavior

```
moveForward()
moveForward()
moveForward()
turnLeft()
turnLeft()
turnLeft()
moveForward()
moveForward()
moveForward()
collectGem()
```

Functions chapter

Creating a New Function

```
func turnRight() {
    turnLeft()
    turnLeft()
    turnLeft()
}

moveForward()
turnLeft()
moveForward()
turnRight()
moveForward()
turnRight()
moveForward()
turnRight()
moveForward()
turnLeft()
moveForward()
toggleSwitch()
```



Learn To Code 1

Functions chapter

Collect, Toggle, Repeat

```
func collectToggle() {
  moveForward()
  collectGem()
  moveForward()
  toggleSwitch()
  moveForward()
}
```

```
collectToggle()
turnLeft()
collectToggle()
moveForward()
turnLeft()
collectToggle()
turnLeft()
collectToggle()
```

Loops chapter

Using Loops

```
for i in 1 ... 5 {
  moveForward()
  moveForward()
  collectGem()
  moveForward()
}
```

Loops chapter

Looping All the Sides

```
for i in 1 ... 4 {
  moveForward()
  collectGem()
  moveForward()
  moveForward()
  moveForward()
  turnRight()
}
```

Loops chapter

To the Edge and Back

```
for i in 1 ... 4 {
  moveForward()
  moveForward()
  toggleSwitch()
  turnLeft()
  turnLeft()
  moveForward()
  moveForward()
  turnLeft()
}
```

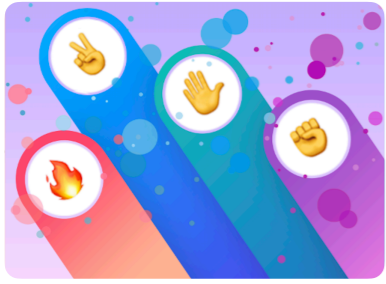


Learn To Code 2

Variables chapter

Keeping Track

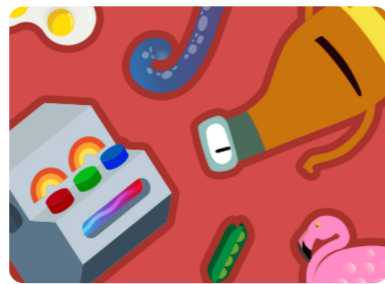
```
var gemCounter = 0
moveForward()
moveForward()
collectGem()
gemCounter += 1
```



Rock, Paper, Scissors

Sample Game

There's no solution example for this page because the game is entirely customizable — you can play it any way you like!



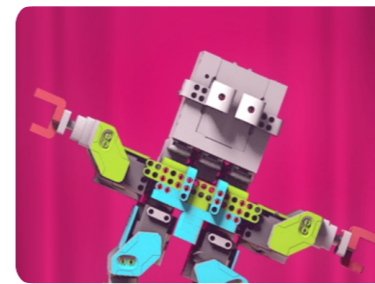
Code Machine

Using a Loop

```
var colors = [Light.red, Light.green,
Light.blue]

var items = [Item.metal, Item.stone,
Item.cloth, Item.dirt, Item.DNA,
Item.spring, Item.wire, Item.egg,
Item.tree, Item.gear, Item.seed,
Item.crystal, Item.mushroom,
Item.unidentifiedLifeForm]

for item in items {
  setItemA(item)
  setItemB(.dirt)
  switchLightOn(.green)
  forgeItems()
}
```



MeeBot Dances

Basic Moves

```
bendAndTwist()
happy()
moveBackward()
shake()
skip()
split()
swagger()
twist()
```

Dance Loops

```
for i in 1 ... 5 {
  bend()
  bend(beats: 2)
  bendAndTwist()
  moveBackward(beats: 9)
}
```

